Making Speculative Scheduling Robust to Incomplete Data

Ana Gainaru[1], **Guillaume Pallez**[2]

1. Vanderbilt University; 2. Inria & Univ Bordeaux;

ScalA@SC19

HPC Schedulers

Reservation-based batch schedulers:

- Relies on (reasonably) accurate runtime estimation from the user/application
- Two queues: (i) large (main) jobs; (ii) small jobs used for backfilling
- **Cost to users**: Pay what you use - need to guarantee that the time asked is sufficient

Reservation-based batch schedulers:

- Relies on (reasonably) accurate runtime estimation from the user/application
- Two queues: (i) large (main) jobs; (ii) small jobs used for backfilling
- **Cost to users**: Pay what you use - need to guarantee that the time asked is sufficient

Under-estimation

- Job killed, need to resubmit; additional cost to user
- Waste of system resources

Reservation-based batch schedulers:

- Relies on (reasonably) accurate runtime estimation from the user/application
- Two queues: (i) large (main) jobs; (ii) small jobs used for backfilling
- **Cost to users**: Pay what you use - need to guarantee that the time asked is sufficient

Over-estimation

- Penalties due to wasting system resources
- Backfilling algorithms might be needed

I have two platforms:

1. Platform A: I pay what I use, **1.5$ per hour**.
2. Platform B: I book some time and pay what I book, **1$ per hour**.

Motivational examples

I have two platforms:

1. Platform A: I pay what I use, **1.5$ per hour**.
2. Platform B: I book some time and pay what I book, **1$ per hour**.

I have one job J_1 whose execution time is **exactly 50h**.

What do you do?

Motivational examples

I have two platforms:

1. Platform A: I pay what I use, **1.5$ per hour**.
2. Platform B: I book some time and pay what I book, **1$ per hour**.

I have one job J_1 whose execution time is **exactly 50h**.

What do you do?

I have one job J_2 whose execution time is **between 46h and 54h**.

What do you do?

Motivational examples

I have two platforms:

1. Platform A: I pay what I use, **1.5$ per hour**.
2. Platform B: I book some time and pay what I book, **1$ per hour**.

I have one job J_1 whose execution time is **exactly 50h**.

What do you do?

I have one job J_2 whose execution time is **between 46h and 54h**.

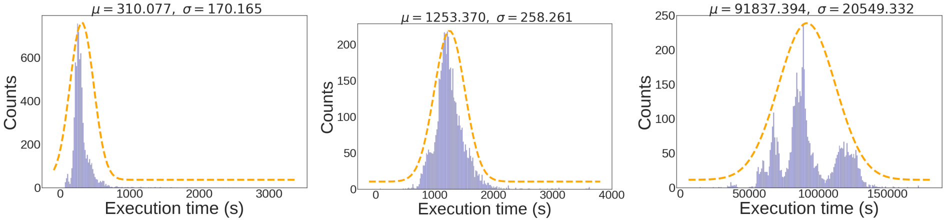
What do you do?

I have one job J_3 whose execution time is **between 2h and 98h**.

What do you do?

Neuroscience Applications

Often the execution time of an application is unknown before it runs.



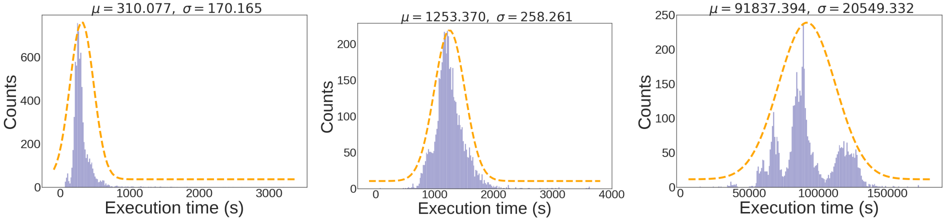
(a) fMRIQA (b) VBMQA (c) dtiQA

Figure: Traces [2013-2016] of neuroscience apps (Vanderbilt's medical imaging database).

These applications are input dependent, but predicting the exact execution time is hard even when knowing the input.

Neuroscience Applications

Often the execution time of an application is unknown before it runs.



(a) fMRIQA (b) VBMQA (c) dtiQA

Figure: Traces [2013-2016] of neuroscience apps (Vanderbilt's medical imaging database).

In our previous work we provide the optimal **sequence of requests** based on: (i) a model of the applications; (ii) a model of the platforms; (iii) resiliency schemes available

Example sequence of requests

For the job with exec **between 2h and 98h**:
(assuming it cannot use checkpointing)

Strategy: t1 = 5h, t2 = 40h, t3 = 60h, t4 = 98h.
If the job is 33h:

1. We run the 5h reservation; it fails
2. **Then** we run the 40h; it succeeds.

Total cost is the cost for both reservations

Example sequence of requests

For the job with exec **between 2h and 98h**:
(assuming it cannot use checkpointing)

Strategy: t1 = 5h, t2 = 40h, t3 = 60h, t4 = 98h.
If the job is 33h:

1. We run the 5h reservation; it fails
2. **Then** we run the 40h; it succeeds.

Total cost is the cost for both reservations

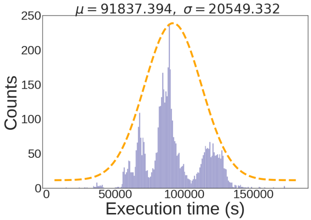
More information in our paper:
Ana Gainaru, Guillaume Pallez, Hongyang Sun, Padma Raghavan: *Speculative Scheduling for Stochastic HPC Applications*. ICPP 2019: 32:1-32:10

Result 3: robustness?

Q: It seems to work, but how do you know the job distribution?

Result 3: robustness?

Q: It seems to work, but how do you know the job distribution?

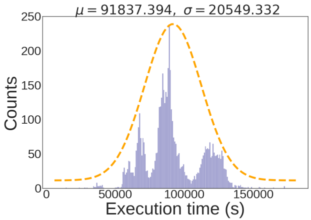


- Dealing with incomplete/low volume of information?
- Here, we consider we know N previous runs for instance.

Result 3: robustness?

Q: It seems to work, but how do you know the job distribution?

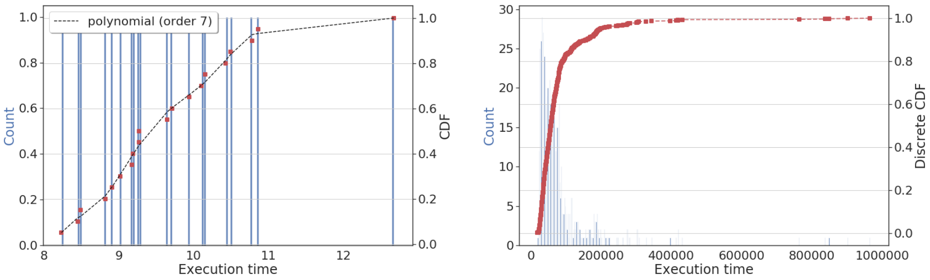
- Dealing with incomplete/low volume of information?
- Here, we consider we know N previous runs for instance.



Reservation strategies

- **Strat. 1 (Discrete):** Use those N elements to compute a reservation strategy;
- **Strat. 2 (Continuous):** Approximate discrete cumulative function with a continuous cumulative function; use this new distribution to compute a reservation strategy.

Computing the discrete CDF

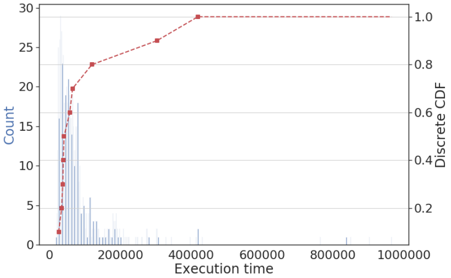


(a) Synthetic (b) Neuroscience

Figure: Use discrete data to define the CDF

Computing the discrete CDF

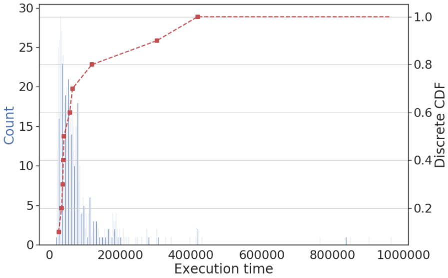
Figure: Using 10 random samples for computing the CDF



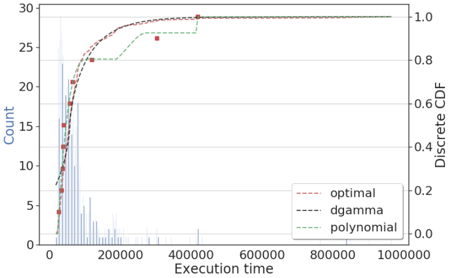
(a) Discrete

Computing the discrete CDF

Figure: Using 10 random samples for computing the CDF



(a) Discrete



(b) Interpolation

Compute the sequence of requests

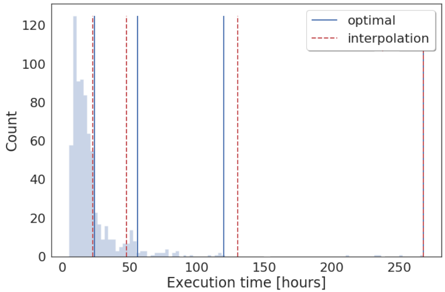


Figure: Example sequence of requests using all data or interpolating 10 random samples

For a job of length t, we define the cost of a reservation t1 as:

$$\textbf{Cost:} \quad = \alpha T + \beta \min(T, t) + \gamma \qquad (1)$$

- Reservation cost: what is paid for the reservation
- Utilization cost: what is paid for the usage
- Setup cost (mix of Reservation/Utilization)

Define the cost of a sequence

The cost for a sequence of requests to the user is:

$$\boxed{C(k,t) = \sum_{i=1}^{k-1} (\alpha t_i + \beta t_i + \gamma) + \alpha t_k + \beta t + \gamma}$$

where k is the smallest index in the sequence such that $t \leq t_k$.

Expected Makespan Loss (EML) The relative cost loss of a sequence compared to the optimal.

Synthetic results

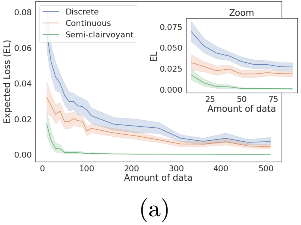
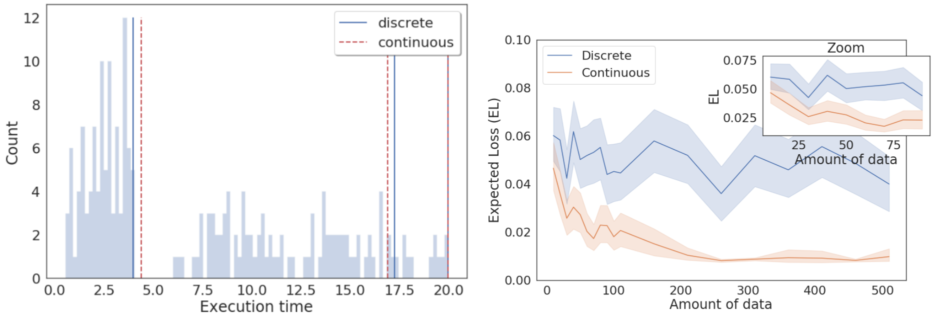


Figure: Results for the truncated normal distributions (left: low variance, middle: high variance) and the truncated exponential distribution (right)

The **Semi-clairvoyant** strategy knows the distributions and only finds the best parameters

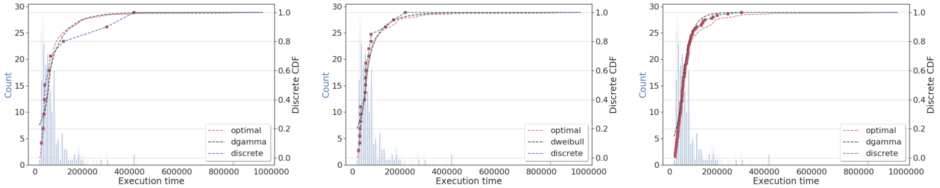
Applications whose historic walltimes have shifts in behavior

Figure: Results for the sum of two truncated normal distributions



(a) Sequences (b) Expected Makespan Loss

Neuroscience applications

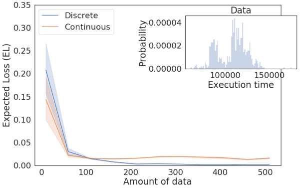


(c) 10 random past runs (d) 20 random past runs (e) 100 random past runs

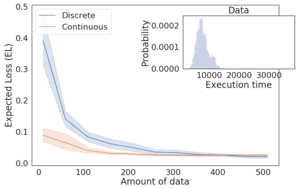
Figure: Using the CDF fit by interpolation or discrete data for different size of training sets

- Continuous fit and discrete data become very similar as we increase the number of samples
- Continuous gives a good fit for the CDF even for 10 datapoints

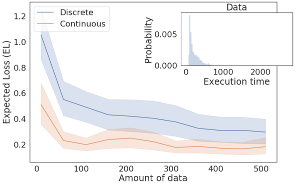
Neuroscience applications



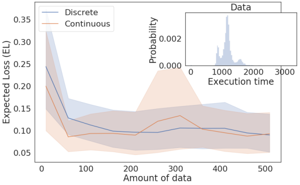
(a) Cerebellum seg (N = 718)



(b) Cortical model (N = 2411)



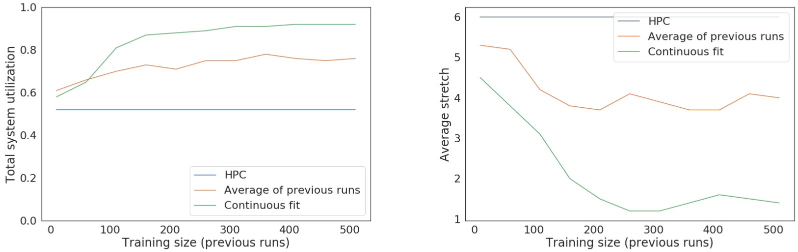
(c) Functional QA (N = 17416)



(d) Deep brain seg (N = 3774)

Figure: Results for real neuroscience applications (Vanderbilt's medical imaging database)

Simulating two weeks of stochastic HPC



(a) System utilization (up is good)

(b) Average job stretch (down is good)

Figure: Running 6 neuroscience applications for a two week timespan

- Average job stretch is improved by 25% when using **only 10 previous runs**
- Training on 100 samples improves utilization by 25%

Conclusions

- Speculative scheduling can be used, even in the presence of incomplete information
- Fitting continuous data seems to show benefits in all studied cases
- Applications do not need many previous runs to start using our scheme

More research is needed

- To understand shifts in behavior from applications
- Design better predictors that can guide our strategy depending on domain knowledge on application behavior